April 2015

Trading in the Financial Market Using Data Mining

Ryan Paul McKenna
Worcester Polytechnic Institute

Trading in the Financial Market

Using Data Mining

A Major Qualifying Project Submitted to the faculty of

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree in Bachelor of Science in

Computer Science

by **Tyler Stone**

and

the Degree in Bachelor of Science in

Electrical and Computer Engineering

by **Ryan McKenna**

4/30/2015

Project Advisors:

Professor Carolina Ruiz
Computer Science

Professor Hossein Hakim
Electrical and Computer Engineering

Professor Michael Radzicki
Social Science and Policy Studies

Abstract

This project developed a replicable process to associate stocks into clusters based on time series data, and selects an appropriate automated trading strategy for each cluster for use in trading. This process included an exploration of data pre-processing methods, selection of a clustering algorithm suited to this application, identification of an optimal investment strategy for each cluster, and the application of strategies on the algorithmically generated portfolio. Efficacy was determined through empirical comparison of gains seen in each test, with the goal of beating the market, or generating percentage greater than the change observed in the S&P 500. This process will serve as a basis for future research and development in the field of applied data mining within the financial domain.

Executive Summary

Background

The proliferation of personal computing in the financial realm has provided investors with unprecedented access to real-time data, allowing for decisions to be made faster and on a broader scope than ever before. To make use of this data, automated trading strategies, which are coded criteria that can signal the purchase or sale of assets to be made when fulfilled, are set up by the user to monitor and trade target assets. The process of determining which assets an investor may focus on is time consuming and non-trivial, relying on extensive domain knowledge. It may be possible, however, to streamline this selection through the application of data mining.

Goals and Objectives

This project aims to develop a process to associate behaviorally similar stocks into portfolios through clustering techniques, and trade the clustered portfolios to beat the performance of the market in terms of percentage gains of the S&P 500 in congruent time frames. Although stock data is analyzed herein, the process used for creation of algorithmically generated portfolios can be used to increase the ease and scope with which an investor can participate in any asset class of the financial market for which time series pricing data is available. In order to accomplish this goal, the following objectives are defined:

- Devise an approach to produce effective clustering of assets.
- Provide a replicable process to implement in a live market
- Evaluate profitability, benchmark against market conditions
- Make recommendations for further process improvement

Methods

A library of Java, Matlab, Tradestation Easy Language, and Python scripts, to be deployed as described in Appendix C, was created to acquire data, preprocess and cluster this data into portfolios, and apply automated trading strategies to said portfolios. Acquisition of data was performed with a Java application that queries the Yahoo! Finance API, creating a CSV file of daily closing prices as a time series for all stocks whose average yearly trade volume is greater than 1 million. Matlab is then used to preprocess each 260-day time series through normalization by Z-score. Matlab then generates clusters from this preprocessed data using the K-Medoids algorithm, outputting a TXT file of cluster centers and all stocks contained in the cluster, as sorted by ascending distance from center.

Five automated trading strategies, selected to perform well in different market conditions, are then applied to the cluster centers in Tradestation. A Python script is used to determine the optimal strategy for each center through a weighted averaging of performance metrics output by Tradestation. A portfolio is then created for each cluster, containing the center and the 20 stocks that are closest to the center. Tradestation's Portfolio Maestro is then used to apply the optimal strategy selected earlier,

to all stocks in the corresponding portfolio. These portfolios were then traded on historical market data in two tests (one over 120 contiguous trading days with portfolios re-clustered at different intervals, and another on five disjoint clustered portfolios spread over five years) to expose the automated trading systems to varied market conditions.

Conclusion

For clustering performed on 260 days of preprocessed closing prices for all stocks, the optimal number of clusters, K, was found to be 10 in most cases. Thus, all clustering was performed for 10 clusters, to enable repeatability and direct comparison of the tests herein. Upon testing, a re-clustering period of 60 days was found to be most viable, performing profitably on average across all market conditions experienced and consistently beating the performance of the S&P 500 in all tests. This suggests that stocks that behave similarly for 260 days will continue to behave within the margins of profitability for the automated trading systems used for 60 days following the end date of the 260-day clustering window, and that a portfolio curated as such should be re-clustered every 60 days to remain profitable. Further research to verify the conclusion of 60-day viability should be performed, as the cumbersome nature of the manual data entry necessary to perform these tests prevented the generation of enough samples to determine the statistical significance of the results. Additionally, refinement of the automated trading strategies used, which were kept simple to limit the variables at play in the tests performed, can greatly augment the profitability of this system for practical implementation.

Acknowledgements

Our team would like to thank our advisors for their support and contributions to our project:

- Professor Carolina Ruiz of the Computer Science Department for sharing her data mining expertise
- Professor Hossein Hakim of the Electrical and Computer Engineering Department for sharing his experience in the financial domain and automated trading systems
- Professor Michael Radzicki of the Social Science and Policy Studies Department for sharing his experience in the financial domain and automated trading systems

Table of Contents

List of Figures

List of Tables

1 Introduction

The financial market, once a vision of instant riches and Wall Street, has become easier to access than ever due to the proliferation of personal computers and the Internet. With online brokering applications such as Tradestation, investors can trade from the comfort of home. Along with this newfound availability of data, automated trading systems have become popular as a means of interpreting and making trading decisions upon market data. A trader can develop a hard-coded trading system that uses the rules and criteria that they had previously followed manually to automatically buy and sell stocks.

While it may seem ideal to automate a computer to make money for an individual, there are many caveats that complicate the process of implementation. For instance, automated trading strategies only perform well under certain market conditions as defined by the user, however the market conditions that the strategy is trading upon can change frequently. An automated trading system alone is not intelligent enough to identify those conditions unless it is programmed to do so. An optimal trading strategy is difficult to quantify, due to the varied conditions under which different strategies perform best, and the differences in what individual investors may seek to reap from the market. The selection of an automated trading strategy to use must be unique to a target stock, or a group of stocks that exhibit the same behavior over time.

This project developed a process by which stocks are grouped into behaviorally similar clusters based on their time series, and traded by automated trading strategies identified to be optimal for each cluster's behavior. This process encompasses extensive research and experimentation, leading to a final process using a Java application for data acquisition, Matlab scripts for data pre-processing and clustering, Tradestation and Python scripts for optimal strategy identification, and Tradestation Portfolio Maestro for backtesting of the portfolios, assessing how the clusters would have traded in historical market conditions. This process will act as a building block for future research and development in applied data mining for the financial domain.

2 Background

2.1 Introduction to the Financial Domain and Portfolio Management

2.1.1 Stock Market Basics

Capital, or the financial reserves used to purchase assets, is the driving force of investment for any entity, from corporations to an individual. Corporations may seek capital to fund infrastructure improvements, new hires, or further investment in other companies. Corporations that opt to be publicly traded have the power to generate capital through the sale of shares in the company to investors in the general public in the form of stocks (Investopedia, LLC). Each stock represents a small sliver of the corporation's equity, represented as a percentage of the company. Thus, when the net worth of a company increases, the value of each share increases accordingly, or decreases if the company loses value over time. Shareholders may opt to buy shares in a corporation that shows promise in increasing in value, or to invest in a company that they may be otherwise invested in the future of. In the case of the individual investor, personal capital may be used to purchase shares in hopes of generating financial gains, thus augmenting the individual's equity.

2.1.2 Stock Portfolios and Trading

The individual assets held by an entity at any given time compose a portfolio. These assets flow in or out of a given portfolio through the purchase, holding, or sale of specific assets, through the transaction of trades in the financial marketplace. Stock assets held in a portfolio can be in the form of a long position, representing that the investor has purchased shares of a corporation in the hopes that they will increase in value to be then be sold at a profit, or a short position, wherein the investor sells shares borrowed from a broker at current value in hopes that the asset will decrease in value, at which point the investor would buy an equivalent number of shares to cover their debts to the broker at the new, lower price (Milton).

2.1.3 Beating the Market

A well curated portfolio should minimally "beat the market," or be more profitable on average than the overall market in comparable time periods. The S&P 500 index is a time series value, reflective of the 500 most valuable stocks in the US, which is commonly accepted as an indicator of market health, making it a good benchmark against which to benchmark the performance of a portfolio. Thus, to beat the market with a portfolio, the curated collection of assets must show gains larger than those of the S&P 500 for the same time period.

2.2 Decision Making in Trading

Due to the vast number of qualitative variables at play, market behavior is non-causal, in that although correlations can be found between different assets and indicators, the Gambler's Fallacy often tricks investors into believing these correlations represent causation. In other words, the individual is led to

believe that events in the current time frame will influence the outcome of random events to come (Phung).

To navigate the complex financial realm, and guide investment decisions, investors rely on many established theorems and rules to make sense of the available data. These rules are often published by established financial institutions, or by successful traders who turn to education and selling knowledge as an additional revenue stream. Generally speaking, these rules and methods are composed of criteria that a stock must meet to be deemed favorable for purchase under different investing schemes. Furthermore, every time a trade is made on a stock, a tick is generated, driving the price up or down by some amount. With roughly 6000 stocks to consider, and approximately 700 billion shares traded per year in the NYSE alone, the full scope of the available data is far more than an individual can monitor (U.S. Census Bureau, 2012). Thus, investors generally limit their scope accordingly, utilizing filters to manage the data, such as looking at weekly values rather than daily values, or participating in a particular sector.

2.2.1 Automated Trading Systems

Automated trading systems, in which strategies are implemented via coded rules that signal the system to buy or sell a given number of shares if a set of conditions is met, offer increased speed and discipline of decision making. Thus, more trades can be made across multiple assets simultaneously than with manual trading, with minimal slippage, defined as "the difference between the expected price of a trade, and the price that the trade executes at" (Investopedia, LLC).

Though a manual trader may use rules to guide oneself in trading, the mind is susceptible to the Gambler's Fallacy or other psychological pitfalls, whereas the deterministic nature of computing allows for coding a set of rules, which will be followed regardless of emotional influence. For example, if a particular position is held, but the value of the asset has declined a significant amount, a manual trader may cave to emotional pressure and withdraw from the position to prevent further losses, closing the trade at a loss. However, the drawdown noticed might be within the realm of statistical possibility, wherein the asset could rebound in value. If the strategy had been well calculated and hard coded into a program, the system would have held onto the position, allowing the value to rebound, making the trade ultimately profitable.

Automated systems are not without pitfalls however, in that the learning curve of implementation may be very steep for individuals unfamiliar with coding, and the conditions under which the algorithms must function is constantly changing with market conditions. Though many tools and pre-coded strategies are available through online brokerage platforms to any individual looking to trade the stock market, the finer points of implementation can be difficult to master, with vast amounts of observation and research necessary to determine when to enter a market and refine a strategy (Wright, 1998). Furthermore, the argument can be made that a well-curated portfolio represents an Ergodic system, wherein the system "forgets" previous states due to ever-changing statistical conditions driving asset values (Veysov, 2012).

2.3 Fundamentals of Cluster Analysis

Automated financial systems provide unprecedented access to data streams and statistical calculation, yet tools must be employed to make sense of this data. One such tool is that of cluster analysis, in which data is grouped in such a way that items within a cluster are similar to one another, but different than those contained in other clusters. In the context of time series data, clustering separates data based on behavioral patterns in the stock. The behavioral similarities found by time series clustering can be visually identified, as shown in Figure 1, which displays a time series clustering of stocks, which can be seen to follow the same upwards and downwards trends over the 260 day period.

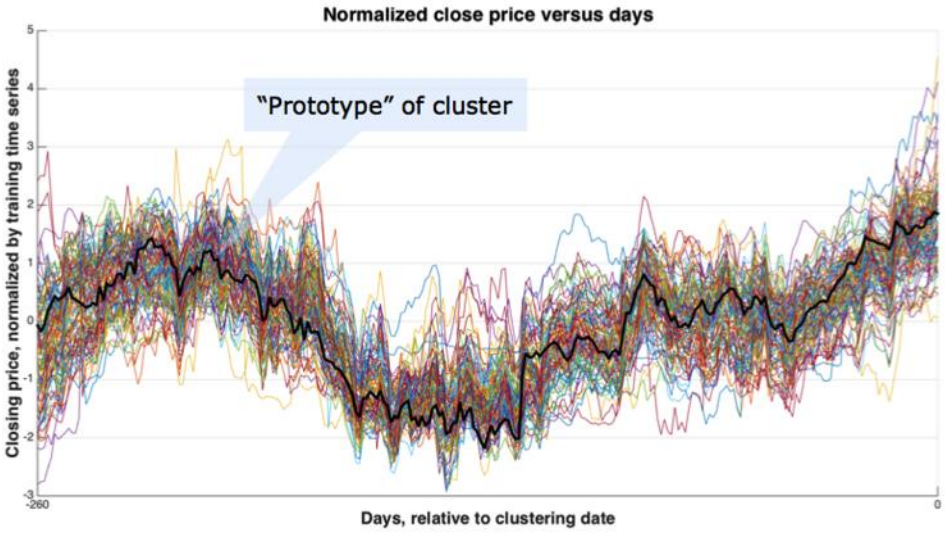

Figure 1 - Stocks clustered over 260 days

The K-means clustering algorithm, of which a variation called K-Medoids was employed herein (as described in Section 3.6.1), is well suited for clustering of time series. Both methods are heuristic, in that their solutions may not be optimal, as demonstrated by the random initial selection of centroids in the algorithm. A hill-climbing method is then employed to iterate toward a more suitable centroid, in which each new cluster is more closely associated by inter-cluster distance (or further associated from other clusters in the case of K-Medoids) than in the previous iteration (Keogh & Lin, 2005).

3 Methodology

3.1 Goal

This project aims to develop a process to associate behaviorally similar stocks into portfolios through clustering techniques, and trade the clustered portfolios to beat the performance of the market in terms of percentage gains of the S&P 500 in congruent time frames. Although stock data is analyzed herein, the process used for creation of algorithmically generated portfolios can be used to increase the ease and scope with which an investor can participate in any asset class of the financial market for which time series pricing data is available.

3.2 Objectives

- Devise an approach to produce effective clustering of assets
- Provide a replicable process to implement in a live market
- Evaluate profitability, benchmark against market conditions
- Make recommendations for further process improvement

3.3 Overview

Figure 2 provides a flow chart of the process used to acquire, pre-process, and cluster asset data for this project. The description below details this same process, as well as the subsequent trading and benchmarking methods.

Data Sample Acquisition:

1. Retrieve data from Yahoo! API *(Java application)*
2. Filter data
3. Select window to cluster upon *(Matlab)*

Clustering:

4. Pre-process data using the Z-Normalize method *(Matlab)*
5. Determine optimal number of clusters by elbow method *(Matlab)*
6. Perform clustering, store cluster results and cluster prototypes *(Matlab)*

Trading:

7. Test upward trending strategy on prototype and export Strategy Performance report to Excel for each cluster *(Tradestation)*
8. Repeat step 7 for downward trending, upward and downward trending, directionless, and volatility strategies

9. Determine the best performing strategy using a weighted average of strategy performance attributes *(Python script)*
10. For each cluster, create a portfolio of the 20 stocks with the shortest Euclidean distance to the prototype, including the prototype *(Portfolio Maestro)*

- Test 1: Repeat steps 4-9 six times at 20 day intervals.
- Perform step 10 to generate 4 portfolios covering 120 days: one clustered six times and traded at 20 day intervals, one clustered three times and traded at 40 day intervals, one clustered twice and traded at 60 day intervals, and one clustered only once and traded for the full 120 days.

- Test 2: Trade using the best performing strategy for each of the portfolio of stocks for 5, 10, 20, 40, 60, and 120 days after the clustered duration *(Portfolio Maestro)*
- Repeat steps 4-10 and for 4 more instances, using disjoint date ranges for clustering, and repeat test 2.

Benchmarking:

11. Compare percentage gains of each test against S&P 500 index (SPY) percentage gains for identical time frames.

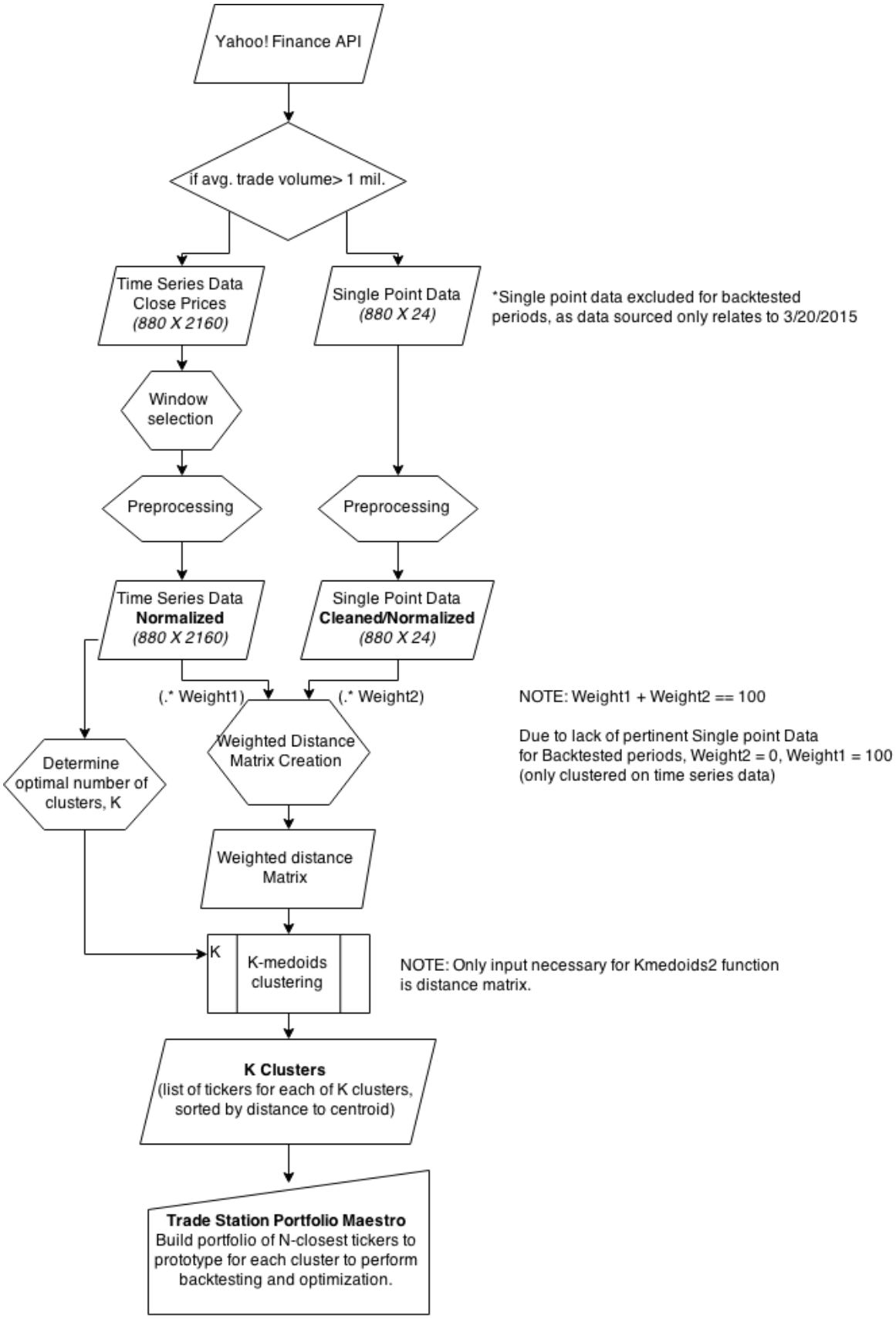

Figure 2 - Flowchart of Clustering Process

3.4 Explanation of Data

3.4.1 Types of Data Acquired

Both single point data and time series data were acquired through a Java script that queries the Yahoo! Finance API. Single point data consisted of fundamental data about a company that is calculated over a longer period of time than one day, such as quarterly or yearly. For a complete list of single point data collected, please see Appendix A. Time series data consisted of the daily closing price of a stock.

The stocks for which data was collected were selected from a vast pool of over 6000 total stocks traded in the NASDAQ, NYSE, and AMEX exchanges. These were limited by means of filtering by single point data and the availability of 2016 days of closing price, as described below in Section 3.4.2. However, due to lack of available single point data relevant to historical dates (such as those used for backtesting), this project was unable to include the acquired single point data in the clustering scheme. Ultimately, the data used for clustering was an 880 by 2016 matrix, representing 8 years of daily closing prices for 880 stocks. 260-day subsets were input to the clustering algorithm (described in Section 3.6). Given a data acquisition date of March 20, 2015, this assumes availability of price data from March 19, 2007 to the date of collection. A complete list of unfiltered and filtered data is available in the github repository listed in Appendix C.

3.4.2 Filtering

This project made use of current single point data for filtering, to eliminate stocks that demonstrate little movement in price. Low volume stocks, with volume being the number of total trades of said stocks, were removed from the dataset by filtering out any stocks with an average yearly volume less than 1,000,000. In order to provide enough data for backtesting, stocks in the dataset were filtered to have at least 2160 days of data. This data represents roughly 10 trading years and allowed for testing on a variety of market conditions. The dataset, after filtering, contained the 2160-day time series for of 880 stocks.

3.5 Data Pre-Processing

In order for the stocks to be compared and clustered accurately, methods for data pre-processing had to be applied to the time series data using Matlab to fit all values to an equivalent range. However, choosing the pre-processing method for this project was not arbitrary. The following methods were tested to determine the best pre-processing technique for the time series data.

3.5.1 De-trending

The process of de-trending the time series data consists of taking the average closing price over a chosen clustering period and subtracting the average from each daily value, removing any upwards or downwards trends (The MathWorks, Inc).

$$\forall\, closing\ price\ \in timeSeries, \qquad detrended\ value = (closing\ price) - (average\ closing\ price)$$

3.5.2 Normalize by Z-Score

Normalizing by z-score is the process in which the average closing price for a given stock over the clustering period is subtracted from each daily value of a given stock and divided by the standard deviation of the closing price for each daily value of a given stock, following the equation below (Wikimedia Foundation, Inc, 2015).

$$\forall\ closing\ price\ \in timeSeries, \qquad Zscore = \frac{(closing\ price) - (average\ closing\ price\ of\ timeSeries)}{(std\ deviation\ of\ closing\ price\ for\ timeSeries)}$$

3.5.3 Scaling

The time series data could be scaled from 0 to 1 by finding the minimum and maximum of each closing price of a given stock. The minimum closing price of the stock is subtracted from each value in the dataset and then divided by the result of subtracting the minimum from the maximum, following the equation below (Wikimedia Foundation, Inc, 2015).

$$\forall\ closing\ price\ \in timeSeries, \qquad scaled\ value = \frac{(closing\ price) - (minimum\ closing\ price)}{(maximum\ closing\ price) - (minimum\ closing\ price)}$$

3.5.4 Choosing the Optimal Pre-Processing Technique

The above pre-processing techniques had to be tested in order to determine the best pre-processing technique or group of techniques to use on the dataset. The techniques were tested using a dataset consisting of 1300 days of 880 stocks. The examined techniques and groups of techniques can be found in **Error! Reference source not found.** below.

Table 1 - Correlation & Sum of squared error of the Clustered Data

	Correlation		Sum of squared error	
Processing Technique	**Random**	**Actual**	**Random**	**Actual**
No Pre-Processing	-0.1320	-0.0230	1004700000	158990000
Detrend Only	-0.1341	-0.0195	974030000	511200
Z-Normalize Only	-0.1557	-0.4523	257050	81562
Scale (0 to 1) Only	-0.1345	-0.4186	23238	7067
Detrend & Z-Normalize	-0.1561	-0.2919	252370	63993
Detrend & Scale (0 to 1)	-0.1372	-0.3120	22502	4152
Z-Normalize & Scale	-0.1338	-0.4255	23258	7002
Scale & Z-Norm & Detrend	-0.1356	-0.3088	22470	4156

First, the optimal number of clusters for the dataset was determined using the sum of squared error elbow method described in Section 3.6.2. Then, the selected pre-processing techniques were applied to the dataset, and the dataset was clustered using the k-means clustering algorithm. The correlation between the proximity matrix and the incidence matrix of the clustering, and the of sum of squared errors of intra-cluster distances were then collected for each test and compared. A proximity matrix is an *n x n* (with *n* being the number of stocks in the dataset to be clustered) representation of the calculated Euclidean distance, described in Section 3.6.1, from every stock in the dataset. For example, the value in row *i*, column *j* of the matrix contains the Euclidean distance between Stock *i* and

Stock *j*. An incidence matrix is an *n x n* matrix of binary values, indicating whether each corresponding pair of stocks belongs to the same cluster. For example, if Stock *i* and Stock *j* belong to the same cluster, the value within the matrix at row *i*, column *j* would contain '1'. If Stock *i* and Stock *j* did not belong to the same cluster, the value within the incidence matrix at row *i*, column *j* would contain '0'. Both correlation and sum of squared error are commonly-used metrics in cluster performance evaluation, and were therefore chosen to evaluate the performance of pre-processing methods. A correlation value of -1.0 is optimal, as it describes clusters that are entirely similar within themselves, and different from other clusters (Tan, Steinbach, & Kumar, 2005). It is important to note that the sum-squared error depends on the input data, as the low sum squared of distances of all scaled trials in Table 1 portrays.

$$SSE = \sum_{k=1}^{K} \sum_{\forall x_i \in C_k} \|x_i - \mu_k\|^2$$

The equation for sum of squared error (SSE) is above, in which there are *K* clusters and x_i represents each individual instance within a cluster. C_k represents the *k-th* cluster. The variable μ_k represents the mean vector of each cluster, represented by the equation below.

$$\mu_k = \frac{1}{N_k} \sum_{\forall x_i \in C_k} x_i$$

In the equation above, N_k represents the number of instances within cluster *k* and each x_i is represented as a vector (Rokach & Maimon, 2010).

The Normalize by Z-score pre-processing method performed best out of the examined pre-processing techniques with the greatest negative correlation, and was thus selected as the sole means of pre-processing our time series data.

3.6 Clustering

After pre-processing, stocks were clustered using the K-Medoids algorithm again using Matlab. Before clustering however, the optimal number of clusters had to be determined, as the number of clusters, *K*, must be specified for the K-Medoids algorithm. For all clustering, 260-day time series (representing roughly one trading year) were used as the input to represent each stock.

3.6.1 K-Medoids

K-Medoids clustering was used to associate stocks based on time series behavior via the Partitioning Around Medoids Algorithm, as described in Figure 3 (Theodoridis & Koutroumbas, 2006).

Partitioning Around Medoids (PAM) Algorithm
1. Initialize: Randomly select *k* of the *n* data points as the medoids
2. Create Clusters: Associate each data point to the closest medoid, *m*, by euclidean distance
3. Evaluate: For each medoid *m*, for each non-medoid point, *q*, swap *m* and *q* and compute new cost (distance) matrix for each configuration
4. Iterate: set configuration with lowest cost as new medoids, *m*
5. Repeat steps 2 through 4 until there is no change in medoids between iterations

Figure 3 - Description of K-Medoids

To determine the clusters, Euclidean distance was used as the distance metric between time series data for a pair of stocks. Euclidean distance is a metric to find the distance between two points. It is simple to apply to multi-dimensional data, making it a common and useful metric. An example of Euclidean distance to find the distance between the time series of stocks A and B with 260 days of data is shown in the equation below (Wikimedia Foundation, Inc, 2015), where A_i and B_i are closing prices on each day for stocks A and B, respectively.

$$Euclidean\ distance = \sqrt{\sum_{i=1}^{260}(A_i - B_i)^2}$$

Although testing with single point data was omitted for this project, a distance metric was implemented to combine the Euclidean distance of time series data and that of the single point data. To use both single point data and time series data in a distance, a weighted average would be applied to the Euclidean distance of time series data (*t*) and the Euclidean distance of the single point data (*s*), following the equation below. The sum of weights (w_1 + w_2) would total 1.0.

$$combined\ distance\ metric = t * w_1 + s * w_2$$

K-Medoids provides an advantage over other clustering algorithms for this project in that the defined centroid of the cluster is an existing data point rather than an averaged point in the feature space, as with k-means. The defined centroid, known as a Prototype, provides a stock, characteristic of the cluster's behavior, upon which to test for an optimal strategy, which can then be applied to the other stocks in the cluster. Setting the seed of the random number generator in Matlab to a fixed value yields consistent cluster centroid selection for comparison, but means that centroids will be found on local optimums rather than global optimum (Keogh & Lin, 2005).

3.6.2 Determining the Number of Clusters

The sum of squared error elbow method was used to determine the optimal number of clusters applicable to the data. The elbow method is a method that iterates from 2 to *n* clusters, finding the average sum of squared error between all the clusters for each iteration. The sum of squared error will

quickly decrease, but as the number of clusters increases the sum of squared error will change in smaller decrements. The "elbow" of this curve is chosen as the optimal number of clusters (Tan, Steinbach, & Kumar, 2005). Figure 4 below illustrates a graph of the sum of squared error and the selected optimal number of clusters, in which there is optimal similarity within clusters, and dissimilarity between clusters.

A script was written to find the optimal number of clusters by from this graph. A line is created, starting at the first point on the graph and connecting to the last point on the graph. Then, the perpendicular distance between this line and each point on the graph is calculated. The point with the largest distance from the line is determined to be the optimal number of clusters.

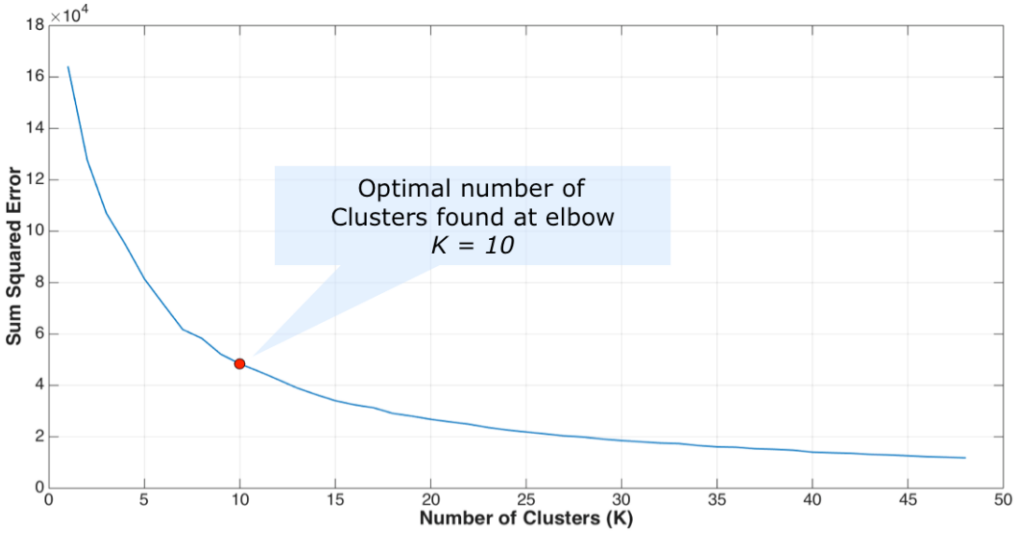

Figure 4 - Sum of squared error per Tested Number of Clusters

The optimal number of clusters for the time series data in the tested time intervals was most commonly 10 clusters. This number was chosen as the default number of clusters for all future work on the dataset.

3.6.3 Presentation of clustering results

To assist in decision-making and the manual entry of clustering results into the trading platform, the clustering results are output to a .csv file, each column representing a cluster. The first element in each column is the prototype of the given cluster, and the successive elements moving down are sorted by distance to the centroid of the cluster. Thus, the closest *n* stocks can be selected for use, omitting outliers that display a high degree of deviance from the prototype, relative to other stocks in the given cluster.

3.7 Trading Systems

Once clusters are established, trading strategies can be applied to cluster prototypes using Tradestation to determine which trading strategies might perform well on the given cluster. A cluster prototype is the centroid of all stocks in a cluster, meaning that its behavior over time is indicative of average behavior witnessed in other stocks of the same cluster. If a prototype stock performs well using a certain strategy,

it is hypothesized that other stocks behaving similar to that prototype will perform just as well with that strategy.

Five simple trading strategies were used. Additional information about these trading strategies, including the Easy Language code for implementation, can be found in Appendix E. The purpose of simple strategies was to prove that the clusters performed well as a portfolio regardless of strategy complexity. The chosen strategies were diverse, therefore performing better or worse depending on the market conditions that a stock is experiencing (Wright, 1998). By applying each of these trading systems to each cluster prototype and comparing the results, the system that performs best can be determined, thus characterizing the cluster.

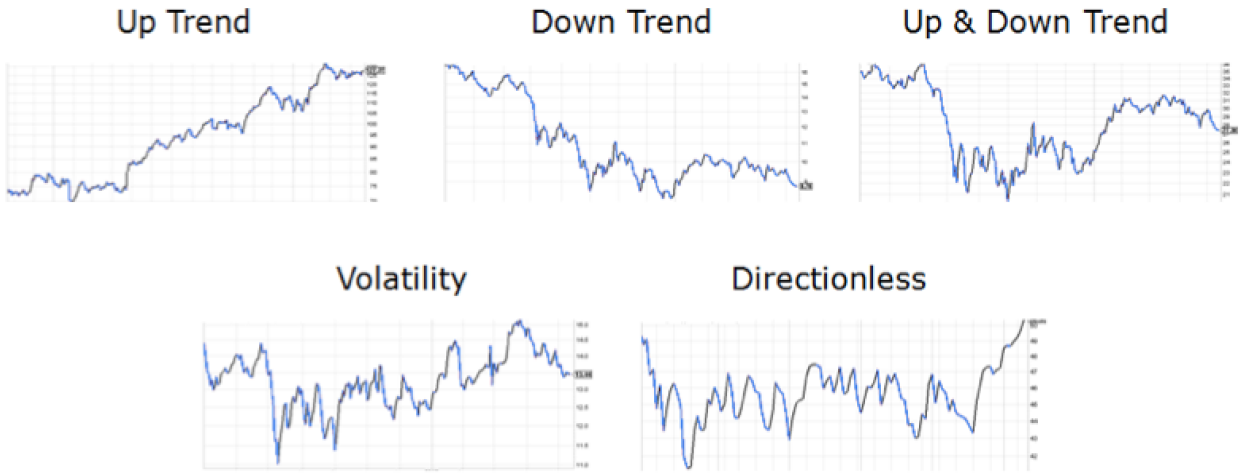

Figure 5 - Automated strategies and respective optimal behaviors (Graphs from Google)

Error! Reference source not found. displays the five trading strategies chosen for application to the clusters, and examples of the respective market conditions that they perform well under. Trend following strategies perform well in market conditions that consistently move upward or downward, showing little deviation from the average rate of change, well suited for trading over long intervals. The up trend and down trend systems developed for this project are very similar, but the downward trending system signals sell short and buy to cover orders instead of buy long and sell orders, as the up trend system signals. In addition, both the upward and downward trend following systems can be applied simultaneously for long term, bi-directional movement. Directionless strategies perform well in market conditions that trade sideways, showing little gain or loss on average across a time period. Similarly, the volatility strategy is profitable in markets that are classified as noisy, capitalizing on the chaotic movements in stock price (Wright, 1998).

3.8 Testing

For the tests performed, backtesting was used in place of real-time testing. In finance, the term backtesting refers to testing a trading strategy over a period of time in history in order to determine whether or not the strategy might perform well in the future. While it would take a long time to test a strategy in real-time, years of past data can be used to benchmark strategy performance and make

informed choices about what strategies may perform best in subsequent days. Tradestation and Tradestation Portfolio Maestro were used for data acquisition and backtesting on market data utilizing daily bars, which each represent the open and close price of a stock over for a given day.

3.8.1 Determining the Optimal Strategy

Using the same time frame upon which each cluster was generated, the optimal strategy for each cluster prototype was tested using a custom Python script. Strategy performance was assessed by selecting eight attributes from the Tradestation Strategy Performance Report that investors commonly use as measures of how well a strategy performs. A weighted average was then applied to the attributes to determine the final "strategy score," with weights set arbitrarily, again according to how an investor may judge efficacy. The attributes and weights can be re-configured to reflect the investor's trading style. An example of this selection process, including the selected attributes and their weights, as well as the weighted average scores for each strategy can be found in Table 2 below.

Table 2 – Example of selected Strategy Performance Attributes and their Weights

Strategy Performance Metric	Weight (%)	Automated Strategies				
		Trend Up	Trend Down	Both Trends	Directionless	Volatility
1 - Net Profit	40%	-$258.00	$322.00	$396.00	-$28.00	$486.00
2 - Profit Factor	20%	0.64	0	2.28	0.91	1.82
3 - Annual Rate of Return	10%	-0.33%	0.41%	0.50%	-0.04%	0.61%
4 - RINA Index	10%	-1.42	2	1.79	-0.22	161.2
5 - # Trades	5%	2	2	4	2	40
6 - Resturn on Investment (ROI)	5%	-0.26%	0.32%	0.40%	-0.03%	0.49%
7 - Return Retracement Ratio	5%	-0.3	0.72	0.69	-0.05	1.39
8 - K-Ratio	5%	0.31	-0.59	-0.2	-0.01	1.97
Weighted average (Strategy Score):		-103	129	159	-11	**213**

The bottom row of Table 2 indicates the "strategy score" for each strategy, as defined in the equation below, which was used to select the optimal trading strategy for the cluster. In this case, the Volatility strategy was selected as it had the most favorable results, shown by the Strategy score being the highest.

$$Weighted\ average\ score = \sum_{i=1}^{n} PerformanceMetric_i * Weight_i$$

3.8.2 Constructing the Portfolio

Once the optimal strategy was selected for each cluster, portfolios were constructed using Portfolio Maestro from the 20 stocks with the shortest Euclidean distance from the prototype, including the prototype. Preliminary testing indicated that portfolios containing the entire cluster contents did not generate profitable results, due to the relative dissimilarity of the behavior of stocks furthest from the prototype in the cluster. This additional step of filtering was performed because K-Medoids clustering classifies all input data into one of the *K* clusters that it generates, leading to the presence of relative outliers within the clusters. Therefore, the number of stocks used in a trading portfolio had to be limited to buffer against the presence of outliers. The number of stocks to include from each cluster into each

portfolio was selected to be 20, as an arbitrary value large enough to show convergent behavior between a group of stocks, but small enough to not capture outliers from clusters that may contain as few as 40 stocks.

3.8.3 Tests

Tests were then carried out on the clustered portfolios using Portfolio Maestro, the results of which can be seen in section 4.1. The optimal strategy for each portfolio was applied on various durations starting at the end of the cluster period. Test 1, of 6x20, 3x40, 2x60, and 1x120 clustering intervals, was performed within a single 120-day period to asses the portfolios' (re-clustered at the described intervals) performance under equivalent market conditions. Test 2 traded 5 disjoint sets of clustered portfolios for 5, 10, 20, 40, 60, and 120 days each, with 5 trading periods for each clustered portfolio, to subject the system to the varied market conditions seen from 2007 to 2012, and assess at which interval the system is most profitable on average. One example of cluster-training and test durations is visualized in Figure 6 below.

Both tests are then benchmarked against the S&P 500 to gauge if the market was successfully beaten in the respective time frame, as seen in section 4.2.

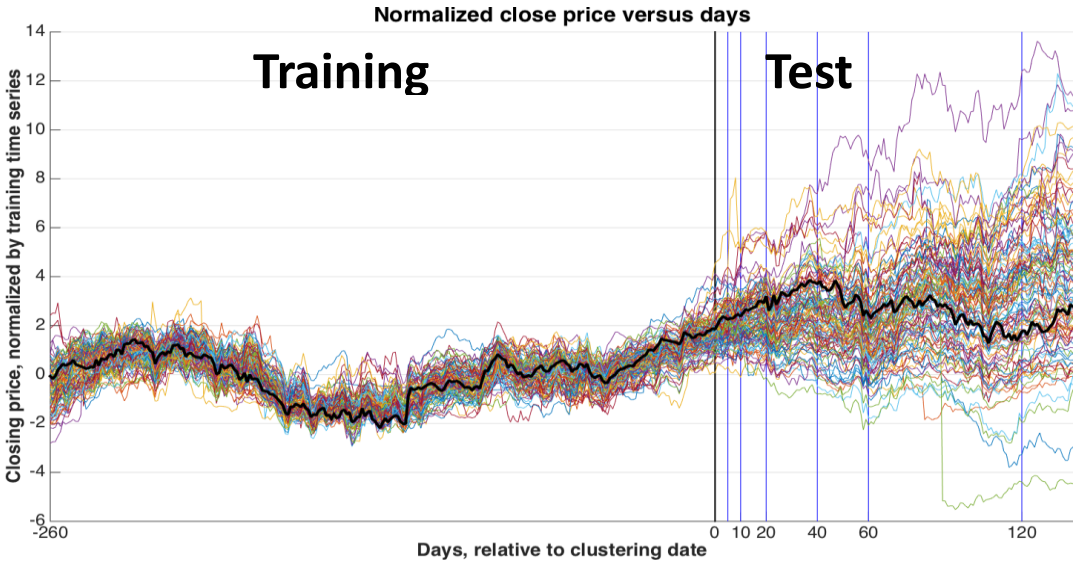

Figure 6 - Visualization of 5, 10, 20, 40 60 and 120 day testing Timeframes

4 Results and Analysis

For reproducibility, all platform information is included in Appendix B, with the exact step-by-step explanation available in Appendix C.

4.1 Backtesting Results

Results of test 1 are shown in Table 3, representing portfolio testing on prototypes and the nearest 20 stocks in clusters generated from the 2012 financial year, and traded during the 2013 financial year with a $100,000 initial investment. The state of the market throughout 2012-2014 has been what is described as a "Bull market", wherein the market is trending upwards consistently. This presents favorable conditions for trading, hence the profitable results across all tested time intervals. The 60-day interval is the most profitable in this test, however, showing 31% gains after trading for 120 days.

Table 3 – 20, 40, 60, and 120-day interval tests for 2013 financial year starting from a $100,000 investment

Cluster Sample:	0	1	2	3	4	5	
	20-day (0)	**20-day (1)**	**20-day (2)**	**20-day (3)**	**20-day (4)**	**20-day (5)**	**Totals**
Start:	10/4/2013	11/1/2013	12/2/2013	12/31/2013	1/30/2014	2/28/2014	
End:	11/1/2013	12/2/2013	12/31/2013	1/30/2014	2/28/2014	3/28/2014	
Profit:	$8,340.00	-$3,641.67	$8,021.00	-$8,311.00	$15,514.00	-$7,021.00	$12,901.33
	40-day (0)		**40-day (2)**		**40-day (4)**		
Start:	10/4/2013		12/2/2013		1/30/2014		
End:	12/2/2013		1/30/2014		3/28/2014		
Profit:	$14,534.00		-$4,105.00		$15,446.00		$25,875.00
	60-day (0)			**60-day (3)**			
Start:	10/4/2013			12/31/2013			
End:	12/31/2013			3/28/2014			
Profit:	$21,693.00			$9,783.00			$31,476.00
	120-day (0)						
Start:	10/4/2013						
End:	3/28/2014						
Profit:	$24,408.00						$24,408.00

The notion that a 60-day interval for re-clustering is most effective is further supported by the results seen in Table 4, which displays the results of test 2. Portfolio information is available in Appendix D. This range of testing dates, selected to include the recession of 2008-2009, presents more diverse market conditions those from which the results in Table 3 were derived. As expected, all tested time intervals, when averaged across each year with an initial investment of $100,000, were far less profitable than results from 2013-2014. However the 60-day test stood out as the only interval that was profitable on average.

Table 4 - 5 year Average Revenue per Trading Interval

INTERVAL		5 day	10 day	20 day	40 day	60 day	120 day
YEAR	2008	-$8,162.00	$2,858.00	-$3,638.00	$12,801.00	$17,847.00	-$41,996.00
	2009	-$13,037.00	-$12,897.00	-$30,026.00	-$38,061.00	-$14,403.00	-$39,659.00
	2010	-$166.00	-$7,825.00	-$14,520.00	-$6,213.00	-$11,151.00	$13,626.00
	2011	$3,498.00	-$3,079.00	$582.00	-$10,656.00	$2,074.00	-$4,443.00
	2012	$9,871.00	$5,314.00	$12,064.00	$25,408.00	$20,666.00	$16,478.00
Average Profit over 5 years:		-$1,599.20	-$3,125.80	-$7,107.60	-$3,344.20	$3,006.60	-$11,198.80

The results above indicate that the portfolios are susceptible to negative market conditions, resulting in a major loss of profit during the recession of 2008. The downturn and reversal of value can be seen in Figure 7, where each clustered portfolio's value is plotted linearly from 2008 to 2012. For comparison, the S&P 500 index's performance for the same time period is included in Figure 8 to show the correlation to the health of the market. As the market rebounded, the net worth of the clustered portfolios increased as well.

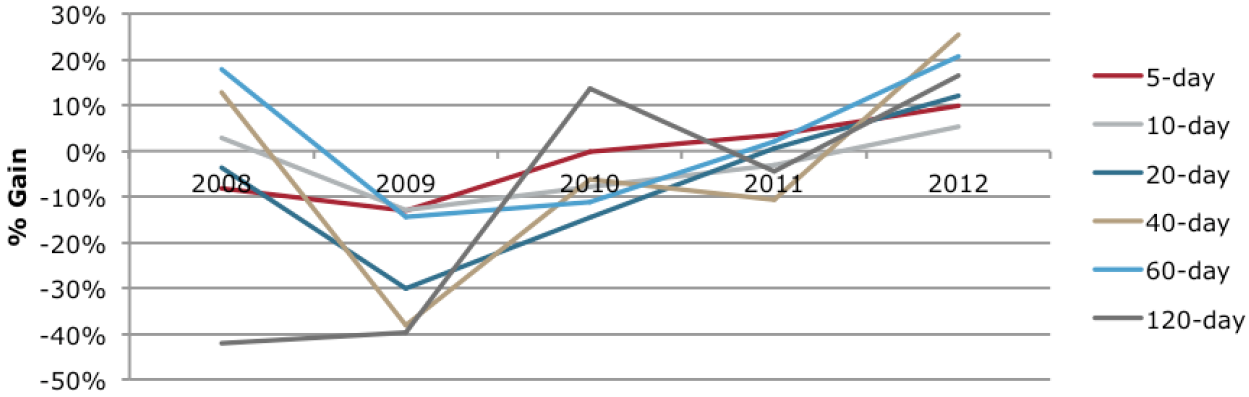

Figure 7 - Net Profit Over Time per Test

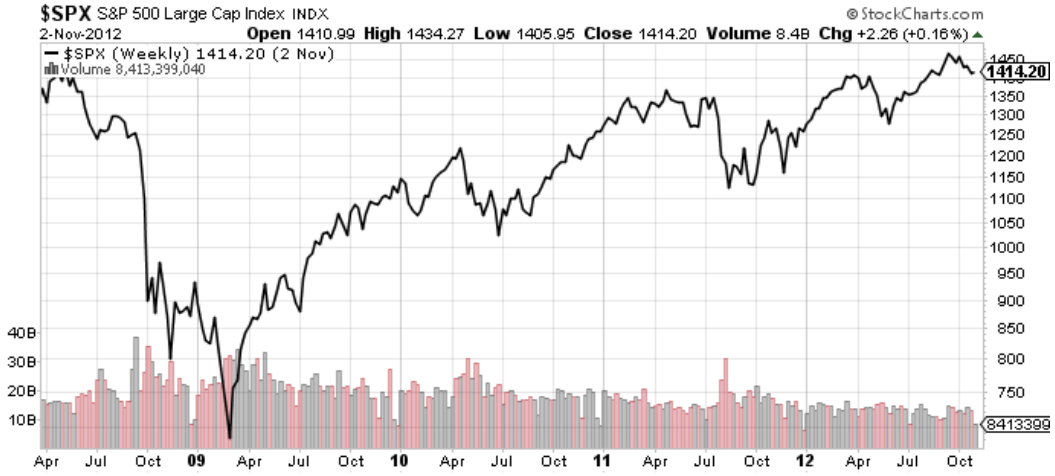

Figure 8 - S&P 500 Index Chart from 4/4/2008 to 11/2/2012

Market conditions also played a major role in the optimal strategy chosen for each cluster, which was expected due to the diversity of conditions favored by each strategy. Figure 9, showing the distribution of strategy selection for each test clustered yearly from 2007 to 2011, illustrates the reduction of the diversity of strategies chosen surrounding the recession. In particular, tests carried out in the years 2008 and 2009 indicated that seven of the ten clusters performed best when using the Trend Down strategy, which would falter as the market subsequently rebounded. More detailed information regarding specific clusters and their associated strategies can be found in Appendix D.

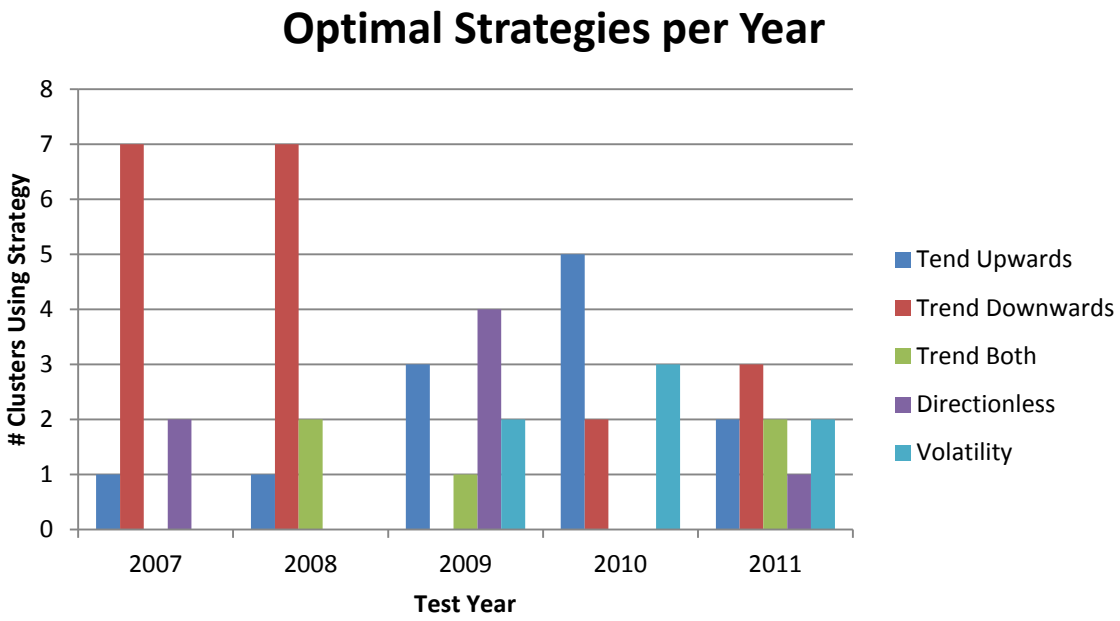

Figure 9 - Optimal Strategies per Year

4.2 Comparison to S&P 500

Test 1, whose results are described in Table 3, represents four portfolios with different re-clustering periods traded on a contiguous 120-day range. Figure 10 portrays the gains of the S&P 500 SPY index and the four clustered portfolios over the 120-day range of October 14, 2013 to March 28, 2014. All clustered portfolios performed above the market, with the 60-day portfolio showing 21.68% gains over the 9.08% change in the SPY index. For this "Bull market" period, all clustered time frames beat the market, with the 60-day portfolio performing best.

Date Range: 10/14/13 - 3/28/14	**Reclustering interval (*number of intervals*)**			
Benchmark: S&P 500 (SPY)	20-day (*6*)	40-Day (*3*)	60-Day (*2*)	120-Day (*1*)
9.80%	12.90%	25.85%	31.48%	24.41%

Figure 10 - S&P and Clustered Portfolio gains for 10/14/2013 to 3/28/2014

Informed by the promising initial results of the 60-day portfolio, additional analysis was performed on the data from Table 4 (test 2) to compare against S&P 500 performance during the same intervals that the clustered portfolio was traded upon. Figure 11 displays the outcome of this analysis, with the 60-day portfolio showing 15% total gains when the SPY index dropped 1.1% over equivalent date ranges, lending more support to the efficacy of the 60-day trading interval.

Time interval	**S&P 500 performance (SPY)**	**60-Day cluster performance**
3/28/08 - 6/23/08	0.45%	17.85%
4/08/09 - 7/06/09	9.42%	-14.40%
4/21/10 - 7/16/10	-11.18%	-11.15%
5/02/11 - 7/27/11	-3.65%	2.10%
5/11/12 - 8/07/12	3.86%	20.60%
Total Gains	**-1.10%**	**15.00%**

Figure 11 - Cumulative S&P 500 and 60-day portfolio gains over 5 years

5 Conclusions and Future Work

Having developed a library of Java, Matlab, Tradestation Easy Language, and Python scripts over the course of this project, recommendations for effective implementation and future academic research must be defined. The above results indicate that although promising in some conditions, room exists for improvement with regards to time series clustering of stock value data with the K-Medoids algorithm. Although statistical significance of both tests conducted cannot be established due to the limited number of tests, these preliminary results indicate that a 60-day trading period may be the most consistently profitable trading window for clusters generated on 260 days of data, having effectively out-performed the S&P 500 in both tests. Further testing of these results is recommended. Furthermore, the use of more robust trading strategies with the inclusion of stop losses and exit strategies can minimize loss seen in periods of downturn in the market, making the proposed system more practical for implementation.

5.1 Need for Improved Data Input Method

Due to the time consuming nature of manual data entry to input clustering results from Matlab into TradeStation, the volume of tests possible to complete in the time available for this project was limited. Testing to find the optimal strategy in TradeStation, and portfolio backtesting in Portfolio Maestro require the manual creation of clusters and strategy application due to a lack of automated interface for rendering the .csv or .txt files of the clusters found by Matlab into portfolios that can be acted upon in these proprietary programs. Thus, testing for the optimal strategy of each of the five options mentioned in Section 3.7 on ten cluster prototypes requires fifty separate strategy performance reports to be generated and then analyzed by a Python script to score the strategies according to the weightings seen in Section 3.8.1. Although initially, the intention was to perform the aforementioned steps over thirty different clustering results, adjacent in time, the process proved too time consuming, thus resorting to an abbreviated test over five different re-clustering periods. The results were not tested for statistical significance due to an insufficient number of samples, and can not be used to make anything more than a cursory deduction and support the need for future testing. Possible future work on the topic should include work on automation of this process to expand the application abilities of Matlab, and improve the ease of performing advanced calculation in TradeStation.

5.2 Improvement of Trading Strategies

The five automated trading strategies (upward trending, downward trending, upward & downward trending, directionless, and volatility) applied are extremely basic, meaning that very few conditions need to be met in order to make a trade. In particular, the strategies are defined in such a way that they are always looking to make trades, employing a single exit condition per strategy. These are criteria that, when met, signal the automated system to terminate any positions held. More robust (and potentially cautious) trading systems would employ multiple exits per strategy, to detect a variety of unfavorable market conditions. The most basic of exits that could be employed to greatly benefit this system's behavior is that of a stop loss, which terminates the position held on a stock when a drawdown of

equity, defined by a dollar amount or percentage, is detected. For example, if 100 stocks are purchased at a total of $1000 with a 10% stop loss, the stocks would be sold if the value of the 100 stocks drops below $900. Such an improvement would improve the resilience of the trading system in conditions such as those seen in the recession of 2008.

6 Bibliography

Investopedia, LLC. (n.d.). *Backtesting*. Retrieved from Investopedia.com: http://www.investopedia.com/terms/b/backtesting.asp

Investopedia, LLC. (n.d.). *Equity Curve*. Retrieved April 3, 2015, from Investopedia.com: http://www.investopedia.com/terms/e/equity-curve.asp

Investopedia, LLC. (n.d.). *Law of Supply and Demand*. Retrieved April 1, 2015, from Investopedia.com: http://www.investopedia.com/terms/l/law-of-supply-demand.asp

Investopedia, LLC. (n.d.). *Public Company*. Retrieved April 1, 2015, from Investopedia.com: http://www.investopedia.com/terms/p/publiccompany.asp

Investopedia, LLC. (n.d.). *Slippage*. Retrieved April 1, 2015, from Investopedia.com: http://www.investopedia.com/terms/s/slippage.asp

Keogh, E., & Lin, J. (2005). Clustering of Time-Series Subsequences is Meaningless: Implications for Previous and Future Research. *Knowledge and Information Systems*, 154-177.

Milton, A. (n.d.). *Long and Short*. Retrieved April 3, 2015, from About.com: http://daytrading.about.com/od/daytradingglossary/g/FuturesLongShor.htm

Phung, A. (n.d.). *Behavioral Finance: Key Concepts - Gambler's Fallacy*. Retrieved April 1, 2015, from Investopedia.com: http://www.investopedia.com/university/behavioral_finance/behavioral7.asp

Rokach, L., & Maimon, O. (2010). Clustering Methods. In *Data Mining and Knowledge Discovery Handbook* (p. 326). New York City: Springer.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining.* New York City: Pearson.

The MathWorks, Inc. (n.d.). *Detrending Data*. Retrieved from Mathworks.com: http://www.mathworks.com/help/matlab/data_analysis/detrending-data.html

The MathWorks, Inc. (n.d.). *kmedoids*. Retrieved from Mathworks.com: http://www.mathworks.com/help/stats/kmedoids.html

Theodoridis, S., & Koutroumbas, K. (2006). *Pattern Recognition 3rd ed.* Waltham: Academic Press.

U.S. Census Bureau. (2012). Statistical Abstract of the United States. 750.

Veysov, A. (2012). *Financial System Classification: From Conventional Dichotomy to a More Modern View.* Munich Personal RePEc Archive.

Wikimedia Foundation, Inc. (2015, March 27). *Euclidean Distance*. Retrieved from Wikipedia.com: http://en.wikipedia.org/wiki/Euclidean_distance

Wikimedia Foundation, Inc. (2015, February 17). *Feature Scaling*. Retrieved from Wikipedia.com: http://en.wikipedia.org/wiki/Feature_scaling

Wikimedia Foundation, Inc. (2015, February 24). *Standard Score*. Retrieved from Wikipedia.com: http://en.wikipedia.org/wiki/Standard_score

Wright, C. F. (1998). *Trading as a Business.* Charlie F. Wright.

Appendix A: Single Point Data

The single point variables available for acquisition from the Yahoo! API are listed below. The only variable used in this project was Average Volume, used during the filtering stage of the process.

- Sector (nominal)
- Market Capitalization (numerical)
- **Average Volume (numerical)**
- 200 Day Moving Average (numerical)
- 50 Day Moving Average (numerical)
- Change from 200 Day Moving Average (numerical)
- Change from 50 Day Moving Average (numerical)
- Earnings Per Share (numerical)
- Estimate EPS Next Quarter (numerical)
- Estimate EPS Next Year (numerical)
- Next Year EPS Price Estimate (numerical)
- Current Year EPS Price Estimate (numerical)
- Year High (numerical)
- Year Low (numerical)
- PEG Ratio (numerical)
- PE Ratio (numerical)
- Short Ratio (numerical)

Appendix B: Platform Information

Matlab	Version	2014b
Tradestation	Version	9.1
Portfolio Maestro	Version	3.0

Appendix C: Detailed Process Breakdown

This appendix references files in the zip file located on the github repository here:

https://github.com/tyler-stone/financial-data-mining

The entire process described in the project is as follows:

Data Acquisition

1. Run the TWNN.jar file under the *data_acquisition* folder and type '1' and press enter. Then, import a CSV file with the list of stocks of interest. The default list used for this project is titled stock_list.csv within the same folder.
2. After, re-run TWNN.jar. This time, choose to retrieve either real-time or historical data. This project only used historical data, but real-time data (or single-point data, as it is referenced within this project) is available to retrieve as well. Choose '4' and press enter, and the application will connect to the Yahoo! API and retrieve all historical data for each stock imported in the database.
3. Re-run TWNN.jar once more, but this time to export either real-time or historical data. Choosing '5,' a CSV file will appear in C:\ with historical data for each stock up to 2016 days. **Note:** this number is equivalent to approximately 5 years of trading data. The data is a constant that can be modified in the source code of the Java project.

Data Pre-Processing and Clustering

4. To load the CSV files generated above into Matlab, include the single point and time series CSV files in the Matlab directory, and call the loadStocks() function, with the following parameters:

    Inputs:

    timeSeriesFilename – [str] name of CSV of raw time series price data for *N* stocks

    singlePointFilename – [str] name of CSV file of single point data for *N* stocks

    Duration – [int] number of days (up to 2160) to include in 'ts' file output

    Outputs:

    ts – array of raw time series data for *N* stocks over duration

    sp – array of single point data for *N* stocks

    tickers – array of *N* stock tickers (symbols, eg: AAPL) corresponding to time series in ts

Example:

```
>> [ts,sp,tickers] =
loadStocks('timeSeriesFilename','singlePointFilename', Duration);
```

5. To pre-process time series data, use the prepWindow() function with the following parameters:

Inputs:

Series – [int array] array of *N* time series to be preprocessed (*ts* from above)

Duration – [int] number of days to preprocess over (260 for all tests)

Offset – [int] number of days between start date of *ts* and the 260 day period to be preprocessed

Outputs:

tsPrep – array of 260 days of normalized time series values for *N* stocks

Example:

```
>> [tsPrep] = prepWindow(ts, 260, offset);
```

6. To evaluate the number of clusters to use (if 10 is no longer valid), use the findOptimalK() function:

Inputs: *tsPrep* – *N* x 260 pre-processed time series

Outputs: *K* – Optimal number of clusters found at elbow of SSE plot

```
>> [K] = findOptimalK(tsPrep)
```

7. To create the weighted distance matrix for use in clustering, use the weighted_multi_distance() function with the following parameters:

Inputs:

sp – [array] single point data (from step 4, above)

tsPrep – [array] pre-processed 260-day time series for all stocks (from step 5)

w1 – [float] weight applied to *sp* in calculating distMat (Equal to 0 in all tests)

w2 – [float] weight applied to *tsPrep* in calculating distMat (Equal to 1 in all tests)

Outputs:

distMat - *N x N* distance matrix

Example:

```
>> [distMat] = weighted_multi_distance(sp, 0, tsPrep, 1);
```

8. To generate clusters based on the distance matrix, call the weightedClustPlot() function with the following parameters:

Inputs:

distMat – [array] *N x N* distance matrix (from step 6, above)

tsPrep - [array] pre-processed 260-day time series for all stocks (from step 5)

numClust – [int] number of clusters, K, to generate (Equal to 10 for all tests)

tickers – [cell array] list of tickers (from step 4)

Outputs:

cVect – *N* x 1 array indicating what cluster each ticker is associated with

midx – 10 x 1 array of indices of cluster centroids

- A file nearest.txt is generated in the directory, listing the stock prototypes for each cluster across the top row, with each column representing a cluster, as sorted low-high by distance to the prototype. This file is what is used as a reference for subsequent tests in Tradestation and Portfolio Maestro.

- A figure displaying all stocks on a single plot, and a plot of each cluster is generated for visual reference.

Example:

```
>> [cVect,midx] =  weightedClustPlot(distMat, tsPrep, 10, tickers);
```

Optimal Strategy Identification

9. Using Tradestation, set the data begin and end date to the same as the days chosen for clustering. Because trading days do not correspond with the calendar, finding the date for data can be difficult. This problem was solved by modifying the Java project (TWNN.jar) source code and re-exporting the date of data point for each symbol rather than the value.

10. For each cluster prototype, backtest each strategy and save the strategy performance report as an excel file with the following name structure: *x*_strat*y*.xlsx, where *x* stands for the cluster number (2-*n* clusters) and *y* stands for the strategy number (2-*n* strategies). In order to identify the optimal strategy, the Python script looks for files that start with the cluster number and then compares it to all other files with the same cluster number.

11. Repeat steps 9-10 for each clustered sample of data. If there is only one set of data that was clustered, there is only one sample.
12. Copy the folder containing the files to the location of the Python script (*identify_optimal_strategy*) and separate each clustered sample into separate folders (i.e. sample0, sample1). If there is only one sample, place all files into a folder labeled sample0 within the same directory structure as the Python script
13. Run the Python script using the command 'python main.py > results.txt'
14. Results.txt should contain the optimal strategy for each cluster.

Trading with Optimal Strategy

15. Using Portfolio Maestro, create a new portfolio and create new strategy groups for each strategy used.
16. For each cluster, create a custom symbol list by clicking 'Add Symbol List' in the strategy group toolbar, and then clicking 'Add/Remove Custom Symbol Lists' in the popup window.
17. Click 'Create' and label the list according to the cluster number. Using the results from step 6, add the first 20 stocks in the column corresponding to the cluster.
18. Repeat step 17 for each cluster.
19. For each strategy group add the symbol lists corresponding to the clusters that performed best under this strategy, using data from step 14.
20. Add each strategy group to the portfolio.
21. Click 'Backtest Portfolio' and choose a begin date that starts on the same day that the clustering period ended, to the desired end date.
22. Backtest the portfolio and view the results of the backtest.

Appendix D: Tested Portfolios 2008-2012

Sample 0				
Cluster Begin	**Cluster End/Test Begin**	**Test**	**Test End**	**Net Profit**
3/19/2007	3/28/2008	5-day	4/4/2008	-$8,162.00
		10-day	4/11/2008	$2,858.00
		20-day	4/25/2008	-$3,638.00
		40-day	5/23/2008	$12,801.00
		60-day	6/23/2008	$17,847.00
		120-day	9/17/2008	-$41,996.00

Cluster #	**Optimal Strategy**	**Cluster Prototype**	**Portfolio**		
0	Trend Downward	LRCX	KLAC	XL	MTZ
			DHT	SNDK	NAT
			ONNN	IDTI	HSBC
			BA	URI	DOV
			ALTR	JBL	LLTC
			RVBD	LUV	LRCX
			SWHC	ZQK	
1	Directionless	GOOGL	GOOGL	CY	USU
			SPWR	CME	JNS
			AAPL	NBG	AET
			SCHW	VOD	MRK
			TROW	BEAV	SAN
			MSFT	DHR	JASO
			DECK	AON	
2	Trend Downward	PNK	PNK	BYD	DRH
			SNV	SHO	S
			MSI	SLXP	AXP
			HIG	CTL	DTE
			ATML	AEG	FCS
			DISH	BEN	GPK
			MRVL	ROVI	
3	Directionless	FCX	FCX	BHP	BBL
			SCCO	FLR	MT
			KBR	DRYS	ABB
			WMB	LUK	XOM
			AME	SBS	PX
			HAL	OII	CNQ
			NOK	BRFS	
4	Trend Downward	SCG	SCG	XEL	WEC

			DUK	OGE	WR
			ED	TE	GXP
			CNP	AEP	CMS
			POM	SNY	AES
			SNE	GENE	ESRX
			MDCO	UNH	
5	Trend Upward	MOS	MOS	MON	BVN
			BMRN	ANR	HES
			CNX	GG	APA
			RRC	SGY	KGC
			DAR	AUXL	ATVI
			FLS	ILMN	ABX
			OI	AEM	
6	Trend Downward	CAR	CAR	COF	ETFC
			C	TRW	FDO
			HBAN	ALU	JAH
			ASNA	LB	BCS
			FITB	RAD	GCI
			TWX	JBLU	OC
			BKD	STI	
7	Trend Downward	NNN	NNN	O	PLD
			WFM	DLR	NKE
			MPEL	HCP	
			NYCB	ALB	
			FNFG	ANF	
			XOMA	HCP	
			CERS	HCN	
8	Trend Downward	HOV	HOV	KBH	PHM
			MAT	UDR	LEN
			TOL	SPF	NRF
			PPHM	SEE	RYL
			CUBE	AIV	SNSS
			BBBY	DHI	FL
			NKTR	SIMG	
9	Trend Downward	LNG	LNG	SPR	IR
			ARMH	ROK	STZ
			GPOR	BWA	BHI
			PWR	ITW	HLX
			LINTA	QTM	RY
			GNTX	GE	A
			GES	DE	

Sample 1

Cluster Begin	Cluster End/Test Begin	Test	Test End	Net Profit
3/28/2008	4/8/2009	5-day	4/16/2009	-$13,037.00
		10-day	4/23/2009	-$12,897.00
		20-day	5/7/2009	-$30,026.00
		40-day	6/5/2009	-$38,061.00
		60-day	7/6/2009	-$14,403.00
		120-day	9/29/2009	-$39,659.00

Cluster #	Optimal Strategy	Cluster Prototype	Portfolio		
0	Trend Downward	ARO	ARO	KND	QCOM
			ARMH	TJX	STJ
			SWKS	ASNA	RVBD
			SYMC	BIG	CRIS
			PMCS	NTRS	AUQ
			CONN	ROST	KSS
			AMZN	DEPO	
1	Trend Downward	CCI	CCI	QQQ	AMT
			WDR	PX	BBRY
			DISH	MRO	INFY
			CRM	ADI	AMAT
			RGP	INTC	IBM
			MXIM	IVZ	JWN
			LLTC	AAPL	
2	Trend Upward	OCN	OCN	FDO	AMGN
			SLXP	RCPI	GERN
			CTRX	APOL	JBLU
			COCO	NFLX	
			GMCR	ALK	
			SQNM	DLTR	
			IMGN	PBCT	
3	Trend Downward	RRD	RRD	PLD	CAT
			CBL	PGH	OMC
			SCI	ERF	MAC
			URI	AIV	MFC
			LUK	MPW	LNC
			EMN	NI	ING
			AXP	WRES	
4	Trend Downward	RL	RL	AMTD	SAP
			LB	URBN	ECL
			SPLS	WU	SCHW

			CVC	AVP	HAS
			FTR	GPS	JAH
			SYY	FAST	PLCM
			LMT	VFC	
5	Trend Upward	SPN	SPN	JOY	TS
			OIS	HK	BTU
			HAL	NOV	ESV
			PTEN	NBR	HLX
			BHI	ACI	KEG
			X	MT	RDC
			CAM	HP	
6	Trend Downward	TC	TC	TER	CBG
			CENX	AKS	CRZO
			GGP	JASO	CBI
			ANF	TEX	TXT
			RIO	VALE	PH
			FBR	SUNE	FCX
			GNW	EBAY	
7	Trend Downward	AINV	AINV	LF	COF
			UDR	WNC	FDX
			HSBC	MAS	DUK
			STM	CMCSK	BAC
			EQR	NWL	AFL
			IP	UNM	HCN
			EL	SYK	
8	Trend Downward	LEN	LEN	AEO	AXL
			RYL	MTG	ICON
			KBH	SNDK	HOV
			MAR	HDB	PENN
			DKS	AGO	CME
			CCE	CMLS	ACAD
			DHI	KMX	
9	Trend Upward & Trend Downward	LNG	LNG	FHN	
			SPPI	WNR	
			UAL	FNF	
			MUX	IAG	
			PPHM	MNKD	
			NKTR		
			MNST		

Sample 2

Cluster Begin	Cluster End/Test Begin	Test	Test End	Net Profit
4/8/2009	4/21/2010	5-day	4/28/2010	-$166.00
		10-day	5/5/2010	-$7,825.00
		20-day	5/19/2010	-$14,520.00
		40-day	6/17/2010	-$6,213.00
		60-day	7/16/2010	-$11,151.00
		120-day	10/11/2010	$13,626.00

Cluster #	Optimal Strategy	Cluster Prototype	Portfolio		
0	Trend Upward	DLR	DLR	BBBY	BC
			JWN	ROK	CTRX
			ETN	ALK	FDX
			BMR	SPG	CCL
			VTR	CRM	IACI
			ALTR	PLD	RMD
			BCE	CNI	
1	Trend Upward	CY	CY	TC	BWA
			VSH	STLD	ALKS
			FCS	CYTR	MTG
			EXAS	ONNN	AMGN
			AGCO	CAB	LVLT
			AKS	FISV	NCR
			MDLZ	RVBD	
2	Volatility	HEB	HEB	FCEL	IRM
			AGEN	FREE	FLR
			ASTI	CTIC	BSX
			ARNA	WFT	ATVI
			SMFG	MTU	AEZS
			XOMA	SUNE	NEE
			GERN	ISIS	
3	Directionless	BEN	BEN	SCCO	OXY
			SGY	L	CSX
			ITW	FCX	CAT
			FLEX	XL	RY
			STZ	FLS	IP
			IR	LEG	RL
			URBN	TYC	
4	Trend Upward	EXK	EXK	SDRL	HK
			HL	ABEV	XCO
			SLW	HAL	IAG

			AG	CEF	AXAS
			EGO	COST	QTM
			ZMH	ABT	KO
			GOOGL	CVC	
5	Directionless	OKE	OKE	XEC	MWE
			OGE	CTSH	TE
			ENB	BTE	ROSE
			TCK	HSP	WLT
			DTE	FOSL	CMS
			INFY	QQQ	WLL
			ARCC	MPW	
6	Directionless	HOT	HOT	FOXA	AINV
			AIV	BEAV	PH
			LB	HST	GNW
			RCL	UDR	TRP
			ALB	MAC	HON
			FOX	PCP	DIS
			EQR	SHO	
7	Volatility	UAL	UAL	SNDK	HD
			HAS	GGP	CLF
			PCYC	AAL	DDD
			NYCB	CREE	SNE
			NOC	FAST	NKTR
			LUV	ASNA	EL
			BIG	JDSU	
8	Directionless	PHM	PHM	SGEN	COCO
			KBH	CVS	PBCT
			TOL	JBLU	
			RYL	NTRS	
			DHI	JEC	
			WEN	MDCO	
			AON	BYD	
9	Trend Downward	LNG	LNG	TCB	APP
			RCPI	SNV	SQNM
			DNR	WNR	GENE
			EAT	BAS	BK
			VLO	MBI	FNF
			SUSQ	NBIX	NOK
			ABC	CLDX	

Sample 3

Cluster Begin	Cluster End/Test Begin	Test	Test End	Net Profit
4/21/2010	5/2/2011	5-day	5/9/2011	$3,498.00
		10-day	5/16/2011	-$3,079.00
		20-day	5/31/2011	$582.00
		40-day	6/28/2011	-$10,656.00
		60-day	7/27/2011	$2,074.00
		120-day	10/20/2011	-$4,443.00

Cluster #	Optimal Strategy	Cluster Prototype	Portfolio		
0	Trend Upward	CAT	CAT	HAL	DD
			JOY	DE	FOSL
			CBI	ALTR	CRZO
			KBR	OIS	ACAS
			ALXN	NVO	EMN
			PAY	BWA	CHKP
			BTE	TRMB	
1	Trend Upward	HES	HES	QQQ	TROW
			SLB	AA	FLR
			VSH	ERF	PAL
			NOV	DOW	PH
			TXN	APA	PLD
			JCI	MXIM	ADI
			HON	DAR	
2	Trend Downward	NKTR	NKTR	ETR	PCYC
			TEF	MRK	BBY
			GSS	WEC	HALO
			ADVS	AVP	MNKD
			ABEV	CSCO	HCBK
			LGF	LF	AGO
			CPB	RAI	
3	Trend Upward	STJ	STJ	WTW	MTZ
			PES	PDS	BAS
			CBS	SSYS	TD
			MTW	SPPI	STO
			CNC	GPOR	NBR
			SGY	CERN	ACHN
			TRN	GLNG	
4	Volatility	SPLS	SPLS	CME	TGT
			MSFT	CMA	INTC
			AMKR	ELX	CYTR

			MGM	CSC	DRYS
			NTRS	TASR	MTG
			ZQK	MRVL	ISIS
			HA	PMCS	
5	Trend Upward	HOT	HOT	AMZN	KMX
			EMR	KIM	BMRN
			HT	UPS	ARCC
			VALE	NSR	EXPD
			RCL	HST	DDR
			PCP	DRH	AIV
			CA	BEAV	
6	Trend Downward	PVA	PVA	UMPQ	ARWR
			GNW	BIOS	CONN
			WY	APP	ARO
			CLRX	VMC	FRO
			CX	BAC	FHN
			PBCT	HPQ	VLY
			COCO	HOV	
7	Volatility	SCCO	SCCO	WIN	BRCM
			CTL	NTAP	FTR
			TV	CCJ	SIMG
			PCAR	F	BAA
			ROVI	AMX	MTOR
			GOOGL	GS	RHT
			LVS	FFIV	
8	Trend Upward	USG	USG	CNQ	SUSQ
			MAS	TOL	MWA
			KBH	PPHM	RAD
			FCEL	RTN	SPF
			DHI	BBT	GPS
			ZION	ODP	RRD
			PPC	NYMT	
9	Volatility	VLO	VLO	OKE	STM
			DVN	TEX	DIS
			TER	GPK	TTWO
			AON	BHI	KND
			WMB	WNR	SM
			LNG	HFC	WFM
			TSO	EXEL	

Sample 4

Cluster Begin	Cluster End/Test Begin	Test	Test End	Net Profit
5/2/2011	5/11/2012	5-day	5/18/2012	$9,871.00
		10-day	5/25/2012	$5,314.00
		20-day	6/11/2012	$12,064.00
		40-day	7/10/2012	$25,408.00
		60-day	8/7/2012	$20,666.00
		120-day	11/2/2012	$16,478.00

Cluster #	Optimal Strategy	Cluster Prototype	Portfolio		
0	Trend Upward & Trend Downward	ALTR	ALTR	JNPR	F
			RCL	ABB	AUO
			SID	VSH	STM
			SEE	RVBD	MS
			FCS	JDSU	HES
			LUV	XRX	ING
			ONNN	GNW	
1	Trend Downward	IAG	IAG	DNDN	WTSL
			KGC	EXC	GFI
			EA	DECK	ABX
			BBG	GG	PEG
			HMY	AU	COG
			END	EGO	GMCR
			DLIA	GST	
2	Trend Downward	AA	AA	ECA	PVA
			GERN	TSL	S
			TLM	ALU	FBR
			JASO	WLT	MT
			NAT	MTOR	CSIQ
			FNFG	ERIC	BHP
			AINV	CENX	
3	Trend Upward & Trend Downward	FTI	FTI	NUAN	GOOGL
			SPPI	NBL	AEE
			OKE	OGE	CAM
			GLNG	MBI	BMRN
			HUM	HCP	MON
			HALO	SBGI	ACHN
			MCD	DUK	
4	Directionless	SPLS	SPLS		
			HLF		
			DDD		

			ENB		
			OII		
			FNSR		
			YHOO		
5	Trend Downward	BTU	BTU	FSLR	XCO
			ACI	PPHM	UPL
			ANR	CNX	SUNE
			BHI	PTEN	CTRP
			NFX	DHT	HAL
			VG	VALE	PGH
			SPWR	FTR	
6	Volatility	A	A	SYK	OI
			TEX	UTX	DOW
			MMM	MET	LXP
			LH	LNC	BKD
			PCAR	BCS	THC
			MTW	HTZ	ALB
			IDTI	CBG	
7	Trend Upward & Trend Downward	STI	STI	IR	HIG
			CMA	PFG	TRW
			BK	JNS	PMCS
			ZION	ZMH	NKTR
			BAC	SNV	HSBC
			MWA	RF	CCK
			SCHW	HCBK	
8	Volatility	HON	HON	SWK	WNC
			LHO	SHO	KEY
			DHR	IVZ	DIS
			ROK	SPR	TWX
			HST	OMC	HOT
			PAYX	PH	TROW
			FLS	BA	
9	Trend Upward	HD	HD	LF	FL
			YUM	LOW	CVS
			PIR	LEN	SPG
			M	DLR	DHI
			URI	PAA	INTC
			USB	ABT	SBUX
			GRMN	KATE	

Appendix E: Trading Strategies

The directionless and volatility strategies used in this project were directly from *Trading as a Business*, referenced in the Bibliography. However, the Trend Upward and Trend Downward strategies were custom strategies developed in EasyLanguage for use in Tradestation and Portfolio Maestro.

Trend Upward

```
inputs:
      Price( Close ),
      Run( 10 );

variables:
      Rise( 0 ),
      AvgWtd ( 0 ),
      Slope( 0 ),
      result( 0 );

begin
      AvgWtd = WAverage( Price, Run );
      Rise = AvgWtd - AvgWtd[Run];
      Slope = Rise / Run;

      Condition1 = Slope > Slope[1] and Slope[1] < Slope[2] and Slope <
0;
      Condition2 = Slope < Slope[1] and Slope[1] > Slope[2] and Slope >
0;

      if MarketPosition = 0 then begin
            If Condition1 then buy at next bar market;
            //If Condition3 then sellshort next bar at market;
      end;

      If MarketPosition = 1 and Condition2 then sell next bar at
market;

end;
```

Trend Downward

```
inputs:
      Price( Close ),
      Run( 10 );

variables:
      Rise( 0 ),
      AvgWtd ( 0 ),
      Slope( 0 ),
      result( 0 );
```

```
begin
      AvgWtd = WAverage( Price, Run );
      Rise = AvgWtd - AvgWtd[Run];
      Slope = Rise / Run;

Condition1 = Slope > Slope[1] and Slope[1] < Slope[2] and Slope < 0;
Condition2 = Slope < Slope[1] and Slope[1] > Slope[2] and Slope > 0;

if MarketPosition = 0 then begin
      If Condition2 then sell Short at next bar at market;
      //If Condition3 then sellshort next bar at market;
end;

If MarketPosition = -1 and Condition1 then Buy to cover next bar at
market;
end;
```